

Remarks

Applicant respectfully requests reconsideration of this application as amended. Claims 1, 8, 13, 17 and 22 have been amended. No claims have been cancelled. Therefore, claims 1-26 are presented for examination.

Claims 1-26 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Archambault (U.S. Patent No. 6,173,444) in view of Blainey (U.S. Patent No. 6,045,585). Applicant submits that the present claims are patentable over any combination of Archambault in view of Blainey.

Archambault discloses a method that reduces the size of alias sets associated with program pointers. First, intraprocedural information about pointer variables referenced in each function of the program is gathered and saved in a data structure called a pointer alias graph. Next, the pointer alias graphs from all the compilation units for the program are combined to form a universal pointer alias graph and then transitive closure is performed on the universal pointer alias graph to produce a reduced graph containing the list of objects that each pointer variable can point to. Finally, all the files are re-compiled using the universal pointer alias graph as input, resolving all occurrences where pointer variables are de-referenced. See Archambault at Abstract.

Standard data flow gathering techniques are used to develop a pointer alias graph. The nodes in the graph represent either a definition of a pointer variable or a use of a pointer variable, and each node has an associated alias set. The initial alias sets for the nodes of the graph are defined as follows: the initial alias set for definition nodes is the right hand side of the pointer variable assignment operation, and the initial alias set for use nodes is the value of the object at that execution point (the r-val). Location information, the basic block number (relative to the flow graph) and position within the basic block, is saved for each node. In order to provide a complete representation of pointer use in the function for the interprocedural analysis, global unique names, called pseudo pointer variables, are assigned

to each formal argument, function return value and global (or file scope) variable.

Corresponding nodes and alias sets are created on the pointer alias graph (col. 5, ll. 4-17).

Applicants submit that Archambault does not disclose or suggest a code segment having a plurality of instructions including a number of pointers wherein at least one of the pointers is a restricted pointer. In fact, the Office Action admits that Archambault does not disclose at least one of the pointers being a restricted pointer. See Office Action at page 2, paragraph 3. However, the Office Action maintains that Blainey discloses such a feature.

Blainey discloses a system and method for determining alias information at a inter-compilation unit level of a compilation process. The method includes the steps of determining anti-alias sets from the alias information provided by a first stage of the compilation process, calculating pessimistic inter-compilation unit alias sets and refining these sets, after transitive closure as appropriate, with the anti-alias sets. See Blainey at Abstract. Nonetheless, Blainey does not disclose or suggest a code segment having a plurality of instructions including a number of pointers wherein at least one of the pointers is a restricted pointer.

Claim 1 of the present application recites receiving a code segment having a plurality of instructions, the code segment having an outer scope and a number of inner scopes, wherein the plurality of instructions comprise a number of pointers, wherein at least one of the number of pointers is a restricted pointer. As discussed above, neither Archambault nor Blainey disclose or suggest such a feature. The Office Action asserts that “restricted pointers are considered a language feature that enables a particular program assertion”. See Office Action at page 3, paragraph 3.

Notwithstanding any validity to the above statement that a restricted pointer is a language feature that enables a particular program assertion, there is absolutely no discussion in Blainey’s described method for determining alias information at a inter-compilation unit

level of a compilation process of the implementation of restricted pointers. In the background section of Blainey there is a disclosure that:

“[I]t is relatively easy to obtain precise aliasing information which is based upon: language rules (eg.--a pointer to a float data type cannot point to an integer data type); language features (eg.--FORTRAN 90's TARGET attribute); or explicit assertions to the compiler made by the programmer (eg.--an ANSI C pragma that specifies that a procedure does not modify external storage locations).

See Blainey at col. 2, ll. 40-46. However, the language features described do not disclose the inclusion of restricted pointers.

Since neither Archambault nor Blainey disclose or suggest receiving a code segment having a plurality of instructions including a number of pointers wherein at least one of the pointers is a restricted pointer, any combination of Archambault nor Blainey would also not disclose or suggest such a feature. Therefore, claim 1 is patentable over Archambault in view of Blainey. Claims 2-7 depend from claim 1 and include additional features. Thus, claims 2-7 are also patentable over Archambault in view of Blainey.

Claim 8 recites receiving a code segment having a plurality of instructions, wherein the plurality of instructions comprise a number of pointers, wherein at least one of the number of pointers is a restricted pointer, and wherein the at least one restricted pointer is in-scope or out-of-scope. Thus, for the reasons described above with respect to claim 1, claim 8 is also patentable over Archambault in view of Blainey. Since claims 9-12 depend from claim 8 and include additional features, claims 9-12 are also patentable over Archambault in view of Blainey.

Claim 13 recites a memory unit to include a code segment having a plurality of instructions, the code segment having an outer scope and a number of inner scopes, wherein the plurality of instructions comprise a number of pointers, wherein at least one of the number of pointers is a restricted pointer. Accordingly, for the reasons described above with

respect to claim 1, claim 13 is also patentable over Archambault in view of Blainey. Because claims 14-16 depend from claim 13 and include additional features, claims 14-16 are also patentable over Archambault in view of Blainey.

Claim 17 recites receiving a code segment having a plurality of instructions, the code segment having an outer scope and a number of inner scopes, wherein the plurality of instructions comprise a number of pointers, wherein at least one of the number of pointers is a restricted pointer. Consequently, for the reasons described above with respect to claim 1, claim 17 is also patentable over Archambault in view of Blainey. Because claims 18-21 depend from claim 17 and include additional features, claims 18-21 are also patentable over Archambault in view of Blainey.

Claim 22 recites receiving a code segment having a plurality of instructions, wherein the plurality of instructions comprise a number of pointers, wherein at least one of the number of pointers is a restricted pointer, and wherein the at least one restricted pointer is in-scope or out-of-scope. Therefore, for the reasons described above with respect to claim 1, claim 22 is also patentable over Archambault in view of Blainey. Since claims 23-26 depend from claim 22 and include additional features, claims 23-26 are also patentable over Archambault in view of Blainey.

Applicant respectfully submits that the rejections have been overcome, and that the claims are in condition for allowance. Accordingly, applicant respectfully requests the rejections be withdrawn and the claims be allowed.


The Examiner is requested to call the undersigned at (303) 740-1980 if there remains any issue with allowance of the case.

Please charge any shortage to our Deposit Account No. 02-2666.

Respectfully submitted,

BLAKELY, SOLOLOFF, TAYLOR & ZAFMAN LLP

Date: May 6, 2004



Mark L. Watson
Reg. No. 46,322

12400 Wilshire Boulevard
7th Floor
Los Angeles, California 90025-1026
(303) 740-1980